

---

## A robust inverse kinematics algorithm for animating a joint chain

---

R. Mukundan

Department of Computer Science and Software Engineering,  
University of Canterbury,  
Christchurch, New Zealand  
E-mail: mukundan@canterbury.ac.nz

**Abstract:** The cyclic coordinate descent (CCD) is a well-known algorithm used for inverse kinematics solutions in applications involving joint chains and moving targets. Even though a CCD algorithm can be easily implemented, it can take a series of iterations before converging to a solution and also generate undesirable joint rotations. This paper presents a novel single-pass algorithm that is fast and eliminates problems associated with improper and large angle rotations. Experimental results are presented to show the performance benefits of the proposed algorithm over CCD and the ‘triangulation’ methods, using different types of cost functions.

**Keywords:** character animation; cyclic coordinate descent; CCD; goal-directed motion; inverse kinematics; IK.

**Reference** to this paper should be made as follows: Mukundan, R. (2009) ‘A robust inverse kinematics algorithm for animating a joint chain’, *Int. J. Computer Applications in Technology*, Vol. 34, No. 4, pp.303–308.

**Biographical notes:** R. Mukundan received his PhD degree from the Indian Institute of Science, Bangalore, India in 1996. He is currently with the Department of Computer Science and Software Engineering at the University of Canterbury in New Zealand. His primary research interests are in the areas of pattern recognition, computer vision and real-time rendering algorithms. He has authored two books and published over 50 papers in journals and conference proceedings.

---

### 1 Introduction

Animation of an articulated structure often requires inverse kinematics (IK) solutions, when only the desired positions of the end-effectors are given. When the number of links in a joint chain becomes greater than three, analytical methods usually become complex and difficult to implement. Iterative numerical methods are therefore commonly used in robotics (Chen et al., 2002) and computer graphics applications (Sumner et al., 2005). An important area where IK algorithms are used is character animation where joint angles of 3D character models are needed to be computed for achieving a goal-directed motion (Bruderlin and Calvert, 1989). Character animation techniques based on motion capture data also require IK solutions for mapping interpolated data to joint positions (Meredith and Maddock, 2005).

One of the well-known algorithms used for computing joint angles from target positions is the cyclic coordinate descent (CCD) method (Lander, 1998). Even though this method can be easily implemented, it has several drawbacks such as the requirement for a number of iterations for certain configurations and undesirable joint angle rotations performed for certain target positions. In order to eliminate some of these problems, a ‘triangulation’ method was recently proposed by Muller-Cajar and Mukundan (2007). This algorithm tries to reduce an  $n$ -link IK problem into a

two-link IK problem using the method of triangulation to reach the target position. However, the main drawback of this method is the need to often rotate a joint by a large angle greater than  $100^\circ$ , which may have to be avoided in many situations with joint angle constraints.

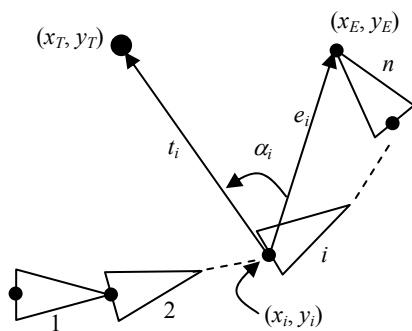
This paper presents an improved version of the triangulation algorithm, designed to provide solutions without large angle rotations. The proposed algorithm is a ‘single-pass’ algorithm in the sense that each link is rotated at most once in an attempt to find a solution. The above characteristics make the proposed algorithm both fast and useful for graphics applications involving multi-joint chains. The paper also presents results of experimental analysis comparing CCD and the triangulation method with the proposed algorithm using different types of cost functions. The paper is organised as follows: the next section gives an overview of the CCD algorithm and outlines its drawbacks. Section 3 gives a description of the triangulation method introduced by Muller-Cajar and Mukundan (2007). Section 4 presents the proposed algorithm. Experimental results are presented in Section 5. Concluding remarks and possible future extensions are discussed in Section 6.

## 2 Cyclic coordinate descent

The author presents below an outline of the CCD algorithm for an  $n$ -link chain as shown in Figure 1, with the following notations:

- $(x_T, y_T)$  position of the target
- $(x_E, y_E)$  position of the end-effectors
- $(x_i, y_i)$  pivot point of the  $i$ th link,  $i = 1, 2, \dots, n$
- $t_i$  target vector for the  $i$ th link  $= (x_T - x_i, y_T - y_i)$
- $e_i$  end-effectors vector for the  $i$ th link  $= (x_E - x_i, y_E - y_i)$
- $\alpha_i$  angle between vectors  $t_i$  and  $v_i$ .

Figure 1 An  $n$ -link joint chain



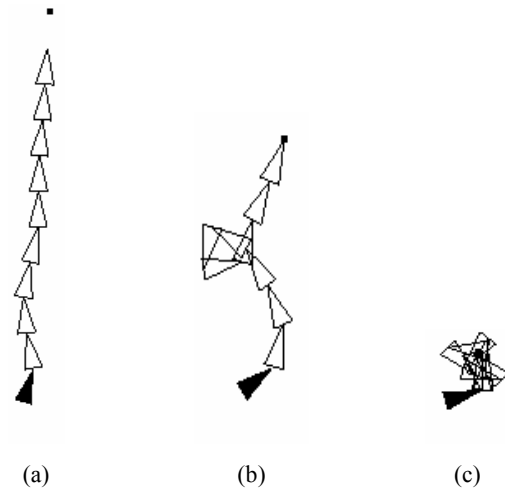
The CCD algorithm can be concisely given as follows:

- 1 Set  $i = n$ .
- 2 Compute  $\alpha_i$ .
- 3 Rotate  $i$ th link by angle  $\alpha_i$  so that the end-effectors meet the target vector  $t_i$ .
- 4 Decrement  $i$  and go to Step 2 if  $i > 0$ .
- 5 Repeat Steps 1 to 4 until target position is reached. We count each repetition of the above steps as one iteration.

CCD's drawbacks are known to the graphics community. Three typical problems are illustrated in Figure 2, using a ten-link joint chain. Throughout this paper, the initial configuration of the joint chain is assumed to be such that the base (triangle with solid colour) is located at the origin and every link is axis aligned with respect to the  $x$ -axis.

For the configuration shown in Figure 2(a), the algorithm requires 100 iterations, though the target could be reached using a single rotation about the base. A simpler solution is possible in the case of Figure 2(b), whereas the CCD algorithm causes the chain to form a loop, intersecting itself. In Figure 2(c), the target position is located close to the base and the joint chain gets crumbled together to reach the target.

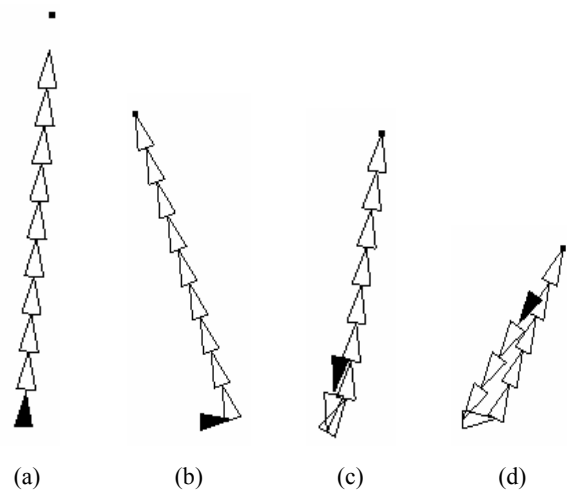
Figure 2 Typical problems associated with CCD algorithm



## 3 Triangulation algorithm

The triangulation algorithm introduced by Muller-Cajar and Mukundan (2007) takes into account the distance of the target from the base and rotates the entire chain (i.e., performs a rotation of the base by angle  $\alpha_1$ ) if the target is not reachable [Figure 3(a)].

Figure 3 The configurations of the joint chain generated by the triangulation algorithm for different target positions



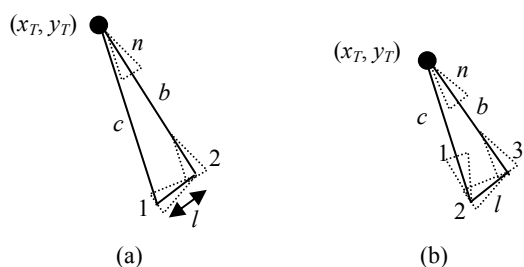
If the target distance is less than the total length of the chain, we have to consider several possibilities. These are explained with the help of the following diagram (Figure 4). Here, we assume that each link has a length  $l$ , so that the total length of the chain is  $nl$ . The distance from the base of a joint chain to the target is denoted by  $c$ .

The triangulation algorithm tries to split the joint chain into two parts consisting of the current link (index 1) of length  $l$  and the remaining links forming a single segment of length  $b$ . As the name implies, the algorithm then tries to form a triangle with  $c$ ,  $l$  and  $b$  as sides so that the end-effectors can reach the target [Figure 4(a)]. The condition for this to be possible is

$$b - l \leq c \leq b + l \tag{1}$$

If the target is close to the base of the joint chain where  $c < b - l$ , then the first link is rotated in the direction opposite to the target vector and is aligned with it [Figure 4(b)], so that  $c$  is effectively increased by  $l$  and  $b$  reduced by  $l$ . If condition (1) is still not satisfied, the next link is also rotated to align with the target vector and the process continues till (1) is satisfied. This situation is illustrated in Figure 3(d). More details about the triangulation algorithm can be found in Muller-Cajar and Mukundan (2007).

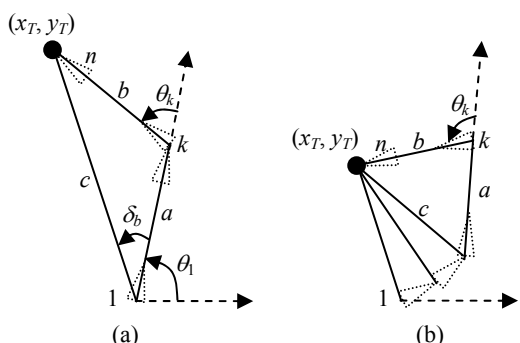
Figure 4 Triangulation algorithm



### 4 Proposed algorithm

The triangulation algorithm obviously performs large angle rotations in order to reach the target. For example, in Figure 3(c), a link is rotated by nearly 165°. Large angle rotations are not acceptable in many situations where joint constraints limit rotations to a maximum value (typically in the range 90°–150°). Using the triangulation algorithm, a target can be approached only from the side of the base, whereas a more natural way to approach a target that is located close to the base is to go around it and try to reach it from the opposite side of the base. The author takes into consideration the above aspects and proposes an improved version of the triangulation algorithm below.

Figure 5 Improvements to the triangulation algorithm



We first try to maximise the minimum angle within the triangle in Figure 4(a), by splitting the total length  $b + l$  evenly. This is done by performing the rotation on a link  $k$  that is closest to the midpoint of the remaining chain [Figure 5(a)]. Thus, we will have the configuration where

the sides of the triangle are  $a, b$  and  $c$ , with condition (1) changed to:

$$|a - b| \leq c \leq a + b \tag{2}$$

The joint angles at nodes with indices 1 and  $k$  are denoted by  $\theta_1$  and  $\theta_k$  respectively and are computed as follows:

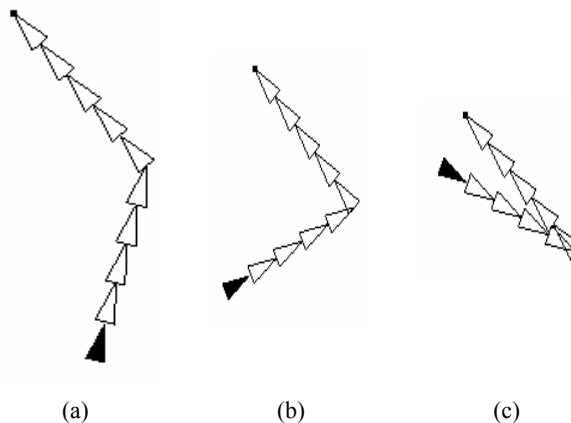
$$\delta_b = \cos^{-1} \left( \frac{a^2 + c^2 - b^2}{2ac} \right)$$

$$\theta_1 = \alpha_1 - \delta_b \tag{3}$$

$$\theta_k = \alpha_k = \pi - \cos^{-1} \left( \frac{a^2 + b^2 - c^2}{2ac} \right)$$

where  $\alpha_i$  is defined as in Section 2. With the above modification of the triangulation algorithm, the results previously shown in Figures 3(b), 3(c) and 3(d) change to that given in Figures 6(a), 6(b) and 6(c), respectively.

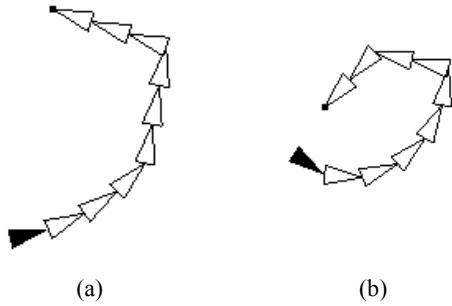
Figure 6 The triangulation algorithm can be modified to split the chain near the midpoint



As seen in Figure 6, the method only produces a two-link equivalent of the joint chain to produce a solution that is devoid of any twisting motion. Large angle rotations are still present, even though some of the unwanted ‘backward’ movement of the chain could be eliminated. The values of  $\theta_k$  in Figures 6(a), 6(b) and 6(c) are respectively 55.4°, 107.21° and 162.71° and the last two configurations are generated by rotations greater than 90°. Therefore, we now consider joint angle constraints and try to avoid rotations that violate such constraints. This can be achieved by orienting the current link at an angle that is nearly orthogonal to the target vector, finding the middle link of the remaining chain, computing  $\theta_k$  and repeating the whole process with the next link if the value of  $\theta_k$  is beyond acceptable limits. This process of ‘going round’ a target is illustrated in Figure 5(b). The actual angle by which we rotate each link should depend on how close or far away the target is with respect to the current link. If the target is too close to the link, we will have to start moving away from the target and later move towards the target. In Figure 7(a), where the target position is same as what is shown in Figure 6(b), the link moves incrementally towards the

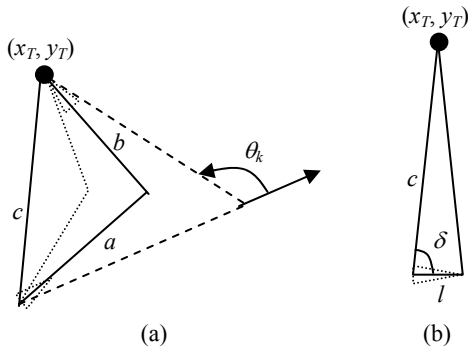
target, till a triangulation with  $\theta_k$  less than  $90^\circ$  becomes possible. In Figure 7(b) [which corresponds to Figure 6(c)], the link is rotated away from the target. The joint angle constraints are met in both cases, with the maximum rotation in the first figure being  $81^\circ$  and in the second figure  $75^\circ$ .

**Figure 7** The proposed algorithm tries to move a link closer or away from the target, depending on its distance from the target



The following figure (Figure 8) explains the important parameters that need to be taken into account while forcing a joint chain to go around a target.

**Figure 8** Angle parameters that control joint rotations in the modified algorithm



With reference to Figure 8(a), the value of  $\theta_k$  is greater than  $90^\circ$  for the outer dotted triangle. Obviously, the necessary condition for this to happen is

$$a^2 + b^2 - c^2 > 0 \tag{4}$$

If the above condition is satisfied, we decide to either move away or towards the target based on the target distance  $c$ . We calculate  $\theta_k$  using (3) and if this angle is greater than  $135^\circ$  in magnitude, we move away from the target, otherwise we move closer. This direction of movement is determined as follows. Referring to Figure 8(b), the distance to the target will not change if:

$$\delta = \cos^{-1} \left( \frac{l}{2c} \right) \tag{5}$$

In order to move closer to the target, we rotate the current link such that it makes an angle  $\delta - 20^\circ$  to the target vector. To move away from the target, this angle is set to  $\delta + 20^\circ$ .

The overall algorithm for the proposed method is given below in pseudo-code form:

- 1 set  $i = 1; k = n/2$
- 2  $a = k * l; b = n * l - a$
- 3 compute distance to target  $c$  from the current link  $i$
- 4 compute  $\alpha_i$
- 5 if  $(c > a + b)$ , then rotate base by angle  $\alpha_i$ ; end
- 6 if  $a^2 + b^2 - c^2 > 0$ , then
  - 6.1 compute  $\theta_k$  using (3)
  - 6.2 compute  $\delta$  using (5)
  - 6.3 if  $\theta_k > 130^\circ$ ,  $\delta = \delta + 20$ ; else  $\delta = \delta - 20$
  - 6.4  $\theta_i = \alpha_i - \delta$
  - 6.5 rotate  $i$ th link by angle  $\theta_i$
  - 6.6  $n = n - 1; k = n/2; a = k * l; b = n * l - a; i = i + 1$
  - 6.7 compute  $a, b$  and  $c$  for the new link; go to 6
- 7 compute  $\alpha_i$
- 8 compute  $\delta_b$  using (3)
- 9  $\theta_i = \alpha_i - \delta_b$
- 10 rotate  $i$ th link by angle  $\theta_i$
- 11 rotate  $k$ th link by angle  $\theta_k$ .

### 5 Comparative analysis

The proposed algorithm is designed to avoid large angle rotations and twisted/self-intersecting configurations that can be produced by CCD and triangulation algorithms. By comparing the pseudo codes of the CCD algorithm in Section 2 and the proposed method in the previous section, three fundamental differences become obvious:

- 1 the CCD algorithm processes links from the end-effectors towards the base, while both the triangulation and the proposed algorithms process links from the base and move towards the end-effectors
- 2 the CCD algorithm uses several passes through the joint chain to converge to a solution, while the proposed method visits each node at most once to find a solution
- 3 the CCD algorithm computes joint angles for every link, while the proposed algorithm rotates only those joints that are needed to move the end-effectors to the target.

The ‘single-pass’ nature of the proposed method makes it a fast algorithm suitable for real-time graphics applications.

The author gives below a comparative analysis of the three methods using different types of cost functions:

- 1 total number of joint angle rotations performed, where only rotations greater than  $5^\circ$  in magnitude are counted
- 2 sum of magnitudes of all joint angle rotations performed
- 3 total distance travelled by the end-effectors.

Ten target positions were randomly generated and the results for the above three cost functions are tabulated in Tables 1, 2 and 3 respectively. As shown in the various examples of the paper, the joint chain had ten links each of

length is two units. The initial configuration of the joint chain for all experiments was parallel to the  $x$ -axis, with the base located at the origin (0, 0) and the end-effectors at (20, 0).

**Table 1** Comparison of the total number of rotations

Target position	Total number of rotations		
	CCD	Triangulation	Proposed
1 (15.17, 4.58)	12	3	2
2 (8.33, 2.83)	9	6	10
3 (-10.58, 2.58)	49	6	4
4 (-4.08, 16.0)	35	2	2
5 (6.41, -10.08)	25	6	6
6 (-17.75, 15.75)	77	1	1
7 (3.33, -13.0)	37	4	3
8 (-1.08, 6.91)	17	8	9
9 (-8.33, 0.33)	34	2	5
10 (0.667, 11.33)	28	6	7

**Table 2** Comparison of the sum of joint angle rotations

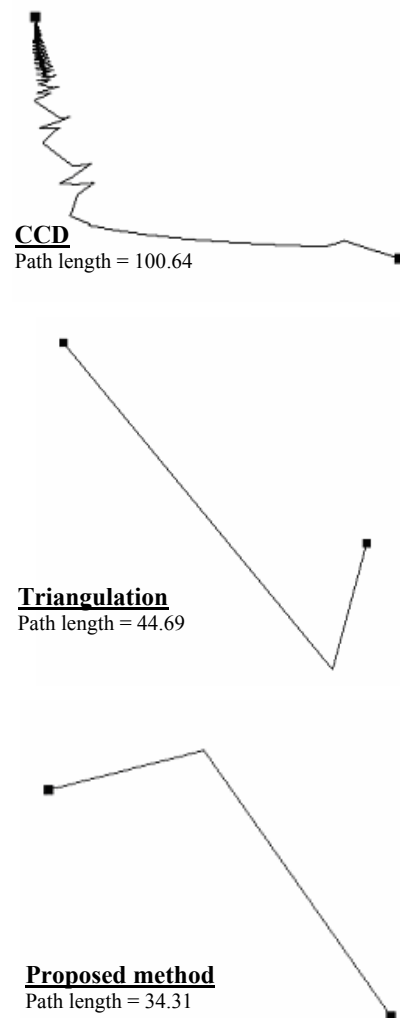
Target position	Sum of joint angle rotations		
	CCD	Triangulation	Proposed
1 (15.17, 4.58)	512.68	355.17	96.09
2 (8.33, 2.83)	668.19	879.56	299.94
3 (-10.58, 2.58)	2,156.80	228.25	225.05
4 (-4.08, 16.0)	1,899.72	171.66	139.43
5 (6.41, -10.08)	914.28	549.08	153.99
6 (-17.75, 15.75)	2,753.46	130.41	138.41
7 (3.33, -13.0)	1,238.17	346.76	119.74
8 (-1.08, 6.91)	1,281.12	593.44	242.92
9 (-8.33, 0.33)	2,067.28	191.73	249.20
10 (0.667, 11.33)	1,350.44	465.85	174.50

**Table 3** Comparison of the distance travelled by end-effectors

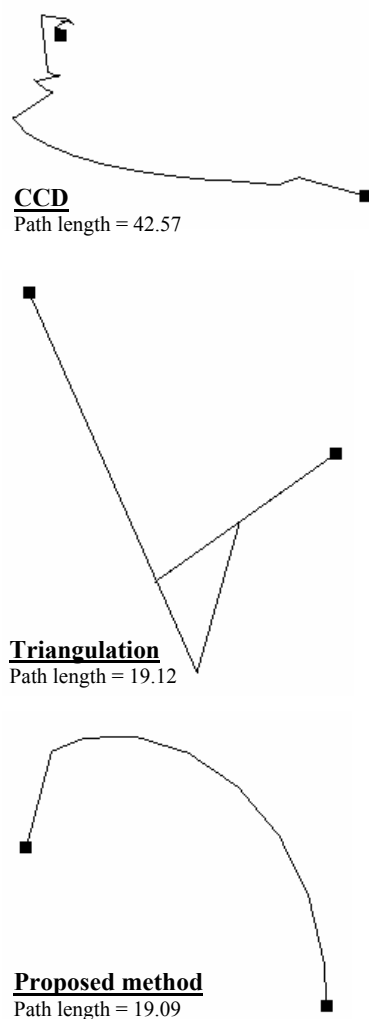
Target position	Total distance traveled		
	CCD	Triangulation	Proposed
1 (15.17, 4.58)	14.05	83.34	19.42
2 (8.33, 2.83)	14.07	192.76	46.35
3 (-10.58, 2.58)	72.81	58.27	52.98
4 (-4.08, 16.0)	100.64	44.69	34.21
5 (6.41, -10.08)	36.80	141.94	26.17
6 (-17.75, 15.75)	207.21	37.39	37.39
7 (3.33, -13.0)	57.72	94.92	25.69
8 (-1.08, 6.91)	31.22	155.56	34.33
9 (-8.33, 0.33)	53.37	39.21	56.95
10 (0.667, 11.33)	40.29	128.15	31.40

From the results presented above in Table 2, it can be seen that the proposed method produces significantly less amount of joint rotations compared to other methods. This is an important cost factor to be considered for both hardware and software implementations as it directly translates to the total effort expended by joint motors. Table 1 shows that both triangulation method and the proposed method generate considerably less number of rotations than CCD algorithm. On an average, the number of rotations for the proposed method is slightly larger than the triangulation method because of the additional transformations used to move around the target for certain configurations. Table 3 shows that the proposed method gives a shorter path for the end-effectors in most of the cases, when compared with the other two methods. Figures 9 and 10 compare the shape and lengths of paths traced of the end-effectors for the three methods and for two different target positions.

**Figure 9** Comparison of end-effectors traces for target position (-4.08, 16.0)



**Figure 10** Comparison of end-effectors traces for target position (3.06, 8.91)



## 6 Concluding remarks

This paper has discussed the IK solution for an  $n$ -link joint chain and the methods used by the CCD algorithm and the triangulation algorithm. The main limitations of the two algorithms have been outlined. The paper then proposed an improved method similar to the triangulation algorithm, but providing a solution without large angle rotations. The proposed method can be easily implemented in real-time rendering applications, as it processes each link at most once to obtain a solution. A detailed comparative analysis has also been presented to show the benefits of the proposed algorithm over CCD and triangulation algorithm in terms of a set of cost functions.

A possible future extension of the method presented is a more general IK solution in 3D space, with quaternion rotations (Aydin and Kucuk, 2006). The solution provided by the proposed algorithm could be further optimised in terms of the cost functions, such as minimum path distance or minimum sum of joint angles.

## References

- Aydin, Y. and Kucuk, S. (2006) 'Quaternion based inverse kinematics for industrial robot manipulators with Euler wrist', *Proc. IEEE Conf. on Mechatronics*, pp.581–586.
- Bruderlin, A. and Calvert, T.W. (1989) 'Goal-directed, dynamic animation of human walking', *Computer Graphics (Siggraph)*, Vol. 23, No. 3, pp.233–242.
- Chen, C.Y., Her, M.G., Hung, Y.C. and Karkoub, M. (2002) 'Approximating a robot inverse kinematics solution using fuzzy logic tuned by genetic algorithms', *Intl. Jnl. of Advanced Manufacturing Technology*, Vol. 20, No. 5, pp.375–380.
- Lander, J. (1998) 'Making kine more flexible', *Game Developer Magazine*, Vol. 11, pp.15–22.
- Meredith, M. and Maddock, S (2005) 'Adapting motion capture data using weighted real-time inverse kinematics', *Computers in Entertainment*, Vol. 3, pp.5–20.
- Muller-Cajar, R. and Mukundan, R. (2007) 'Triangulation – a new algorithm for inverse kinematics', *Proc. Image and Vision Computing – IVCNZ 07*, Waikato, New Zealand, 5–7 December 2007, pp.181–186.
- Sumner, R.W., Zwicker, M., Gotsman, C. and Popovic, J. (2005) 'Mesh based inverse kinematics', *ACM Trans. on Graphics*, Vol. 24, No. 3, pp.488–495.